

# OneEye\_DAS\_Quickstart\_1 for KIT\_AURIX\_TC334\_LK

Read / write variables over DAS using OneEye

AURIX™ TC3xx Microcontroller Training  
V1.0.0



[Please read the Important Notice and Warnings at the end of this document](#)

## Scope of work

---

**Demonstrate how to use the OneEye DAS interface to access variables.**

After configuring the OneEye DAS interface, OneEye is used to read / write variables.

# Introduction

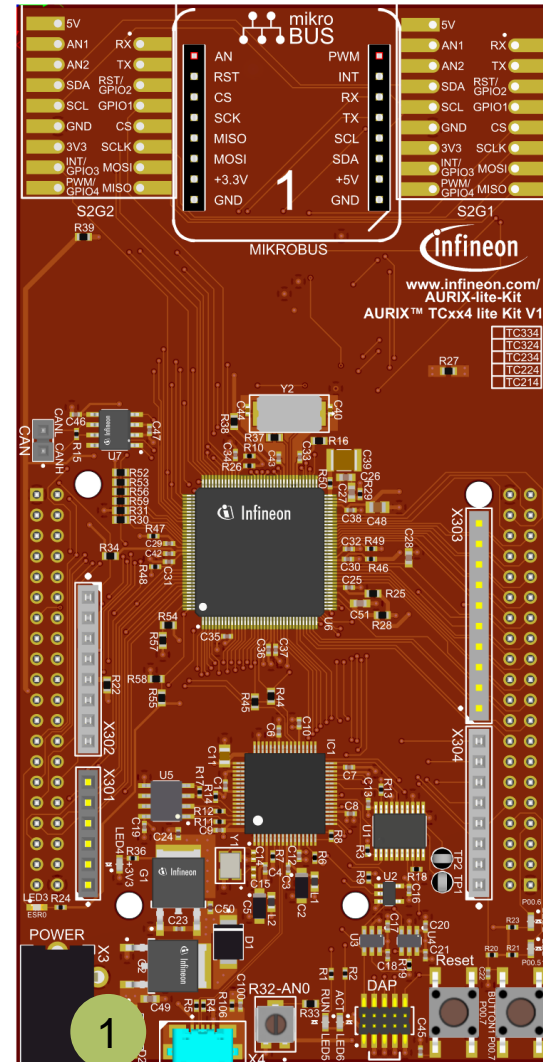
---

- › **OneEye** is a GUI that enables the creation of interactive Graphical User Interface. Graphical elements can be drag from a toolbox and drop onto the GUI. The behavior of the created GUI can be customized. Different communication interfaces like UART, Ethernet, CAN, DAS can be used to interact with the embedded system
- › The **DAS** (Device Access Server) can be used in line with Infineon Microcontroller Starter Kits, Application Kits and DAP MiniWiggler to access the micro controller resources
- › **Recommendation:** It is recommended to go through some of the **basic tutorials** listed in the help embedded in OneEye (Menu: Help -> OneEye help). This enables a quicker ramp-up in the OneEye concept and ensure a nice journey with OneEye

# Hardware setup

This code example has been developed for the board KIT\_A2G\_TC334\_LITE.

The board should be connected to the PC through the USB port 1



# Implementation - AURIX

---

## Configuring the signal generator

A signal generator is used to provide the user with some value to read / write. The signal generator does nothing more than incrementing two signals **signalA** and **signalB** stored in the structure **g\_signalGenerator** up to a maximum before resetting them.

The initialization of the signal generator is done with **initSignals()**.

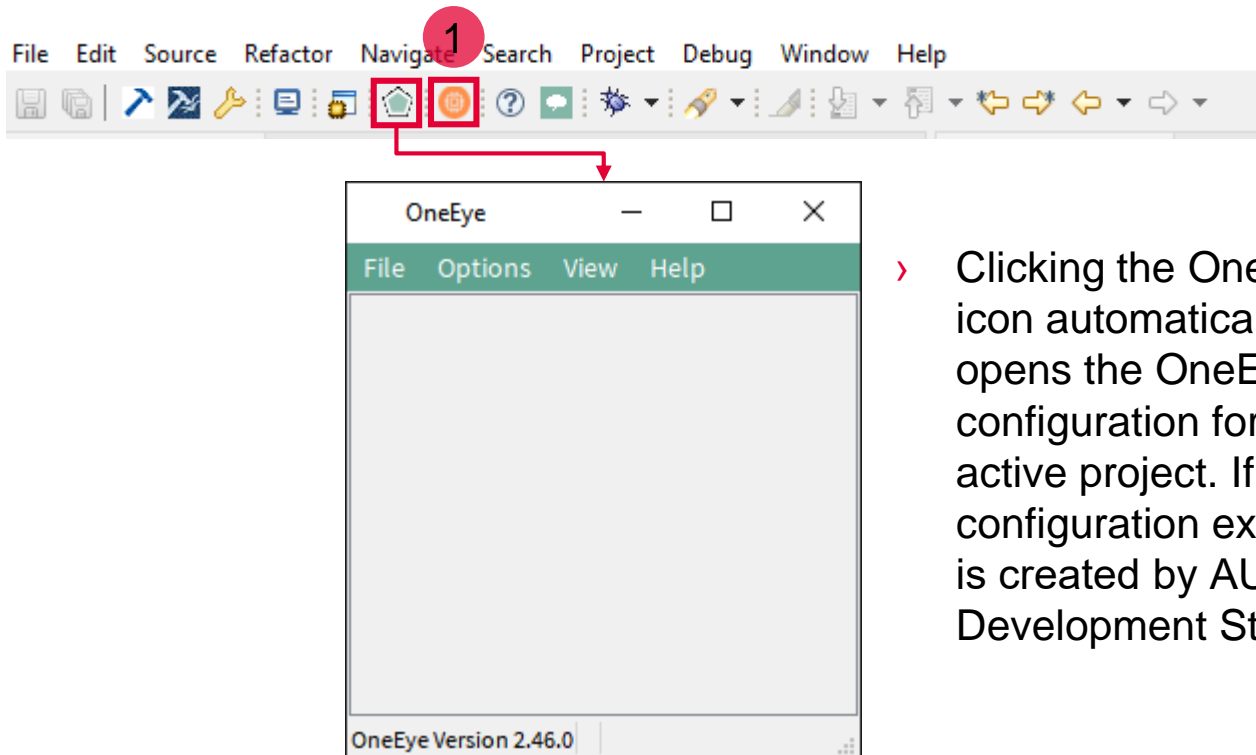
## Running the signal generator

The signal generator is executed in the background loop every 1ms with **computeSignals()**. For that a **deadline** variable is initialized with **getDeadline()** and periodically updated with **addTTime()** to obtain the 1ms period.

**Note:** The access to software variable does not require any special code or library, as the DAS enables OneEye to directly read / write the microcontroller memory.

# Run and Test

- › After code compilation, flash the device using the Flash button **1** to ensure that the program is running on the device
- › For this training, the OneEye application is required for visualizing the values. OneEye can be opened inside the AURIX™ Development Studio using the following icon:



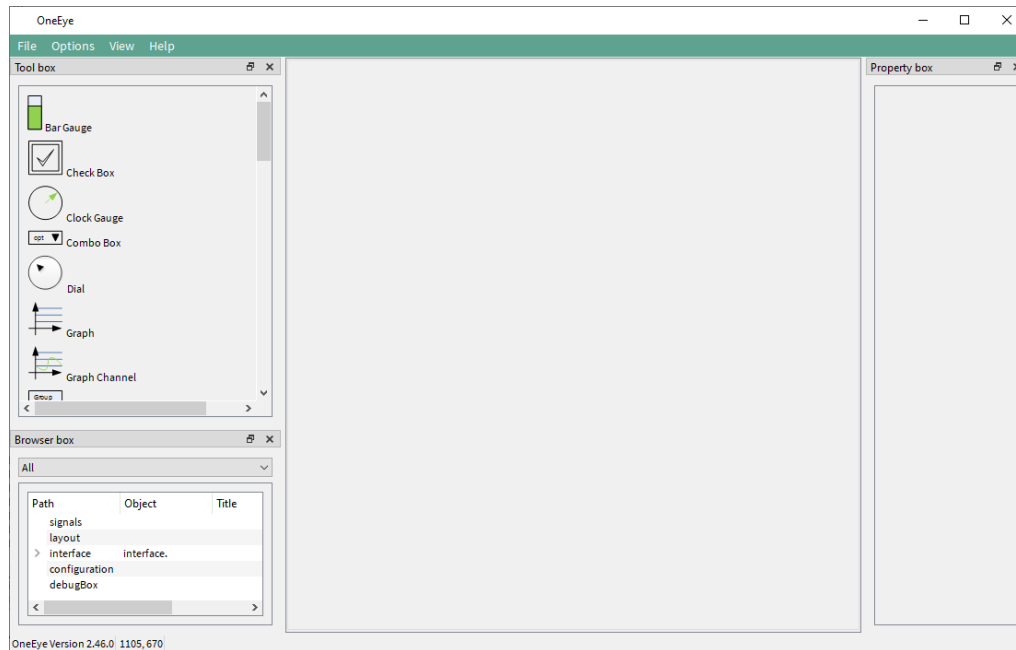
- › Clicking the OneEye icon automatically opens the OneEye configuration for the active project. If no configuration exists, it is created by AURIX™ Development Studio

# Implementation - OneEye

In this training, the OneEye configuration is provided inside the Libraries folder. The following steps are needed to configure the OneEye from a brand-new configuration.

## Setup OneEye for editing

Select the OneEye menu **“Options -> Edit mode”** (if not already checked) to enable the edit mode.  
 Select the OneEye menu **“View -> Browser box”**, **“View -> Property box”**, **“View -> Tool box”** (if not already checked) to display the browser, property box, and tool box.  
 Close the Welcome screen if it was shown.

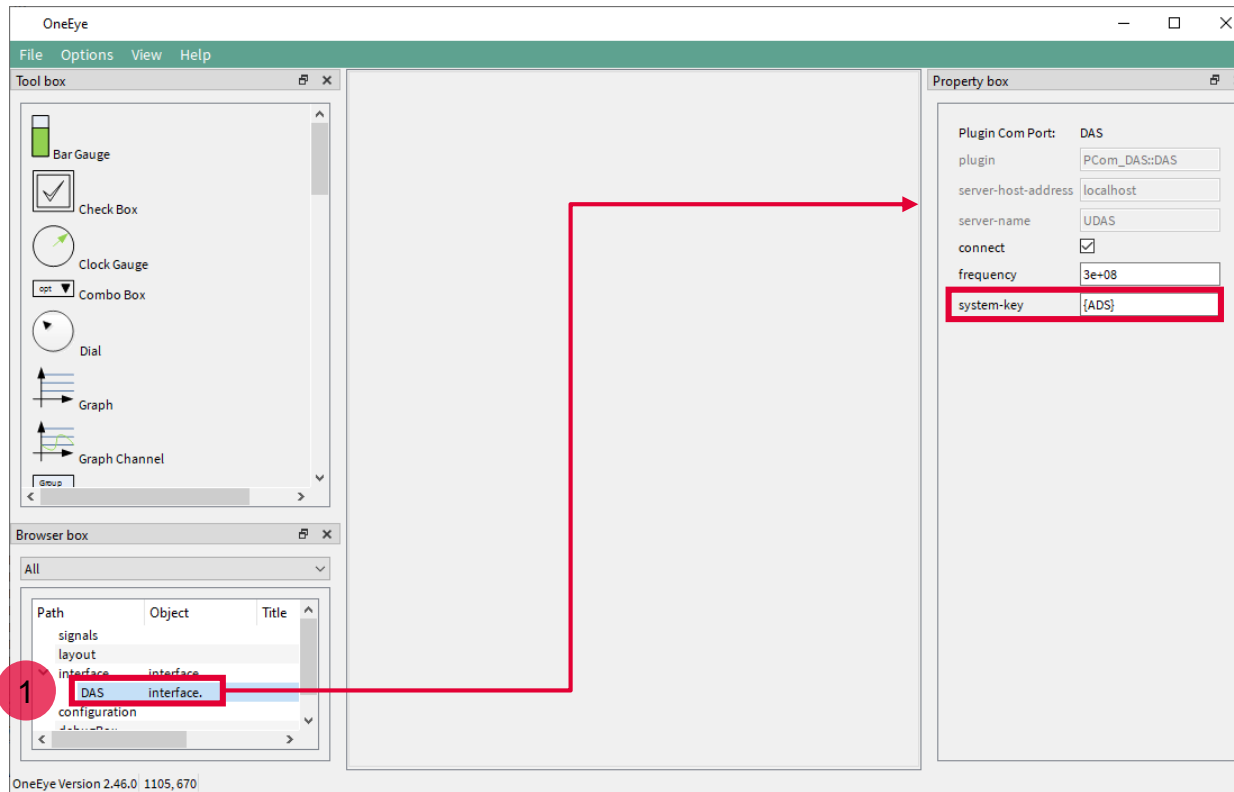


# Implementation - OneEye

## Configuring the DAS interface

When the OneEye configuration is created by ADS, it is already setup with a DAS interface. Select the DAS interface in the Browser box **1**.

Notice the “system-key” **{ADS}** that enables the connection to the device in parallel with the ADS debugger.





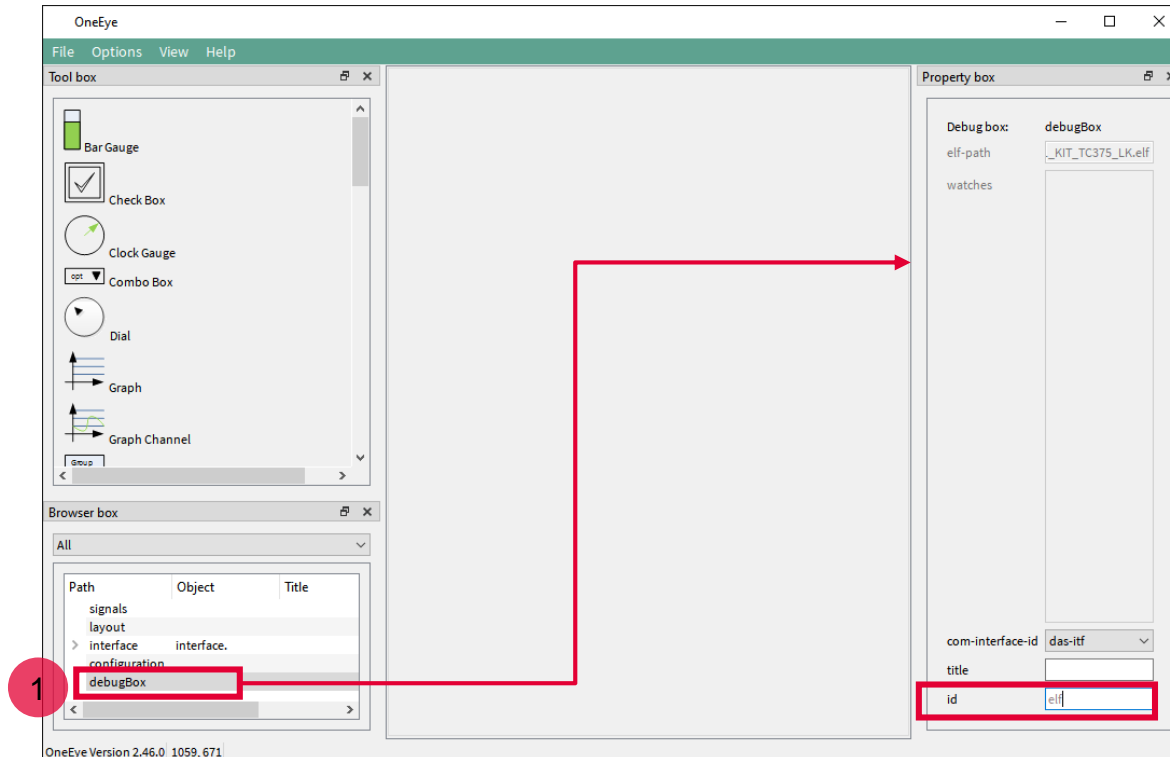
# Implementation - OneEye

## Create a debug box to get access to variables from the .elf file

A debugBox item is already setup by default when ADS creates the OneEye configuration, preconfigured with the project .elf file path.

Select the DAS interface in the Browser box **1**.

Set the id property to “elf”, which enables to group variables into the signal tree later.



# Implementation - OneEye

## Open the debug box viewer and connect to the device

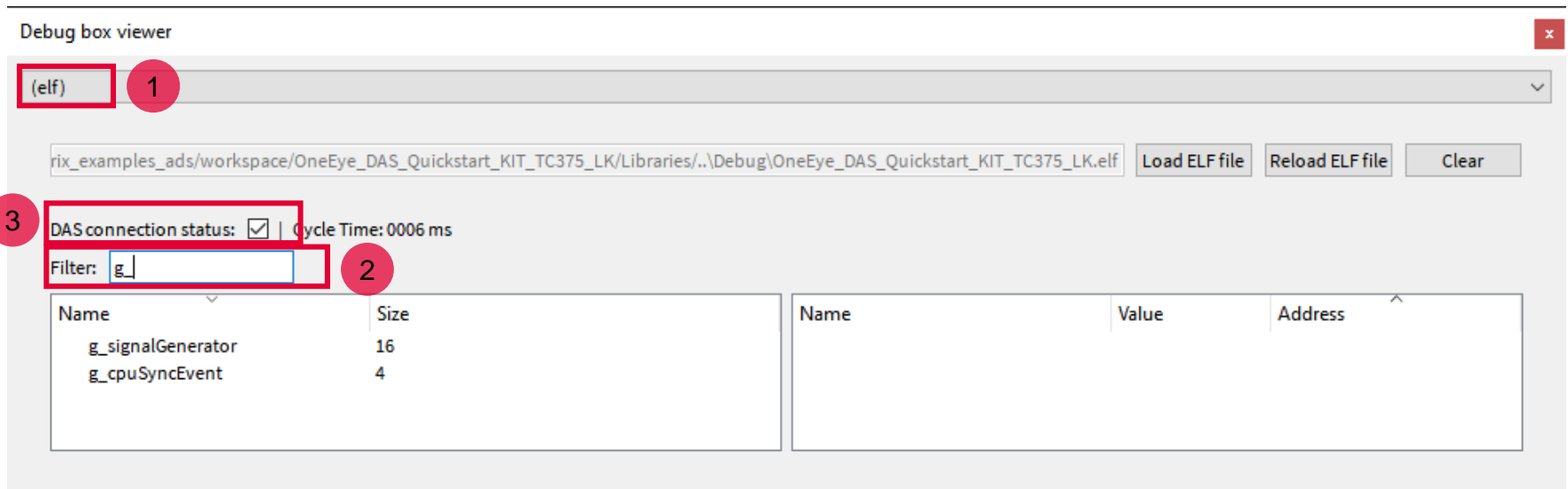
Select the OneEye menu “**View -> Debug box viewer**” (if not already checked) to display the debug box.

Select the debug box with the id “(elf)” **1** if not yet selected by default.

Note that the debug box enables the selection of the .elf file to be used to get information about the variables.

The **Filter** field **2**, enables to filter variables by name. E.g. in this example, entering “g\_” will filter for global variables.

To enable the connection with the microcontroller and have read / write access to variables, check the “**DAS connection status**” box **3**.



Debug box viewer

(elf) **1**

rix\_examples\_ads/workspace/OneEye\_DAS\_Quickstart\_KIT\_TC375\_LK/Libraries/./Debug/OneEye\_DAS\_Quickstart\_KIT\_TC375\_LK.elf Load ELF file Reload ELF file Clear

**3** DAS connection status:  | Cycle Time: 0006 ms

Filter: g\_ **2**

Name	Size
g_signalGenerator	16
g_cpuSyncEvent	4

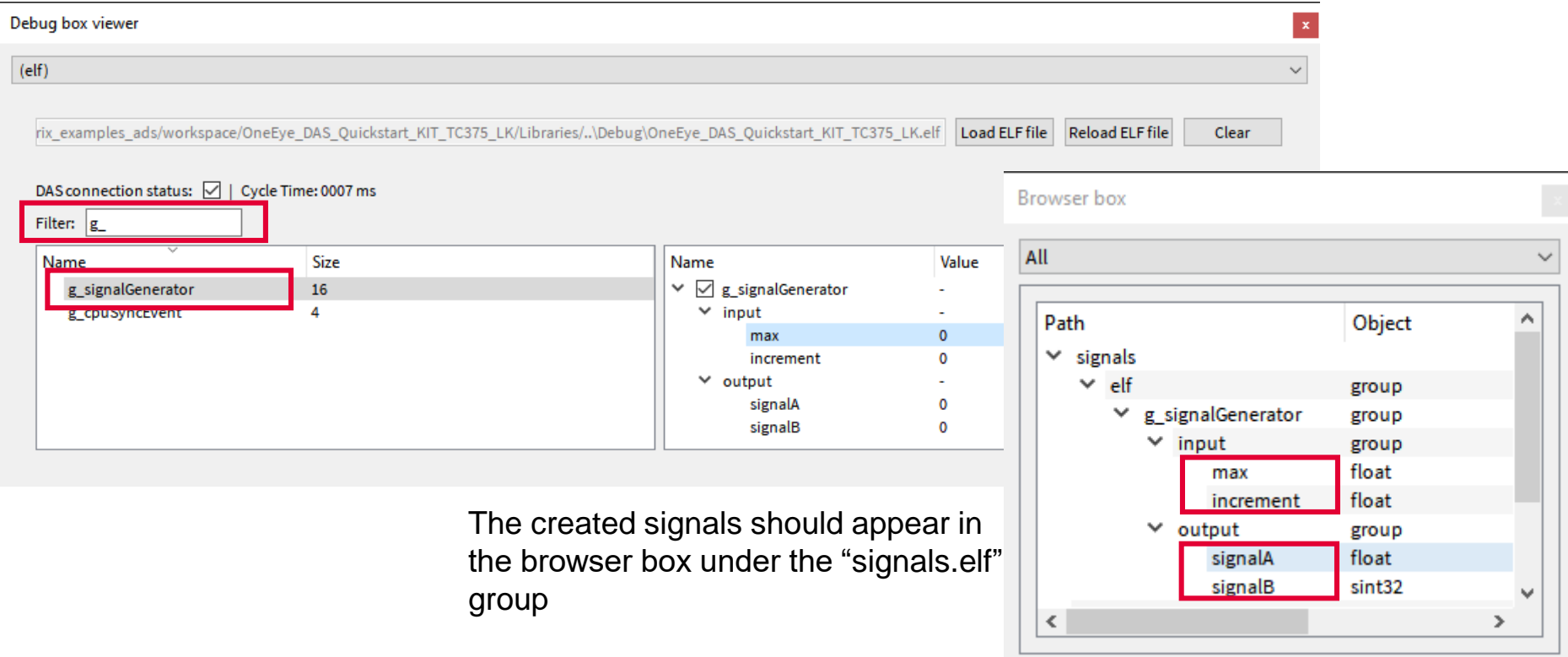
Name	Value	Address
------	-------	---------

# Implementation - OneEye

## Create signals for variable

In the debug box, search for the ***g\_signalGenerator*** variable, right click on it and select “**Add watch on: g\_signalGenerator**”. The watch should appear on the right side of the debug box. Watches are periodically polled for new values on the micro controller.

Expand the ***g\_signalGenerator*** item on the right and right-click on each watch member variables to create a signal with “Create signal for: ...”



The screenshot shows the 'Debug box viewer' window with the following components:

- Filter:** `g_`
- Variable List:**

Name	Size
<code>g_signalGenerator</code>	16
<code>g_cpufreqevent</code>	4
- Watch List:**

Name	Value
✓ <code>g_signalGenerator</code>	-
<input type="checkbox"/> <code>input</code>	-
<code>max</code>	0
<code>increment</code>	0
<input type="checkbox"/> <code>output</code>	-
<code>signalA</code>	0
<code>signalB</code>	0

The 'Browser box' window shows the following structure:

Path	Object
signals	group
elf	group
g_signalGenerator	group
input	group
max	float
increment	float
output	group
signalA	float
signalB	sint32

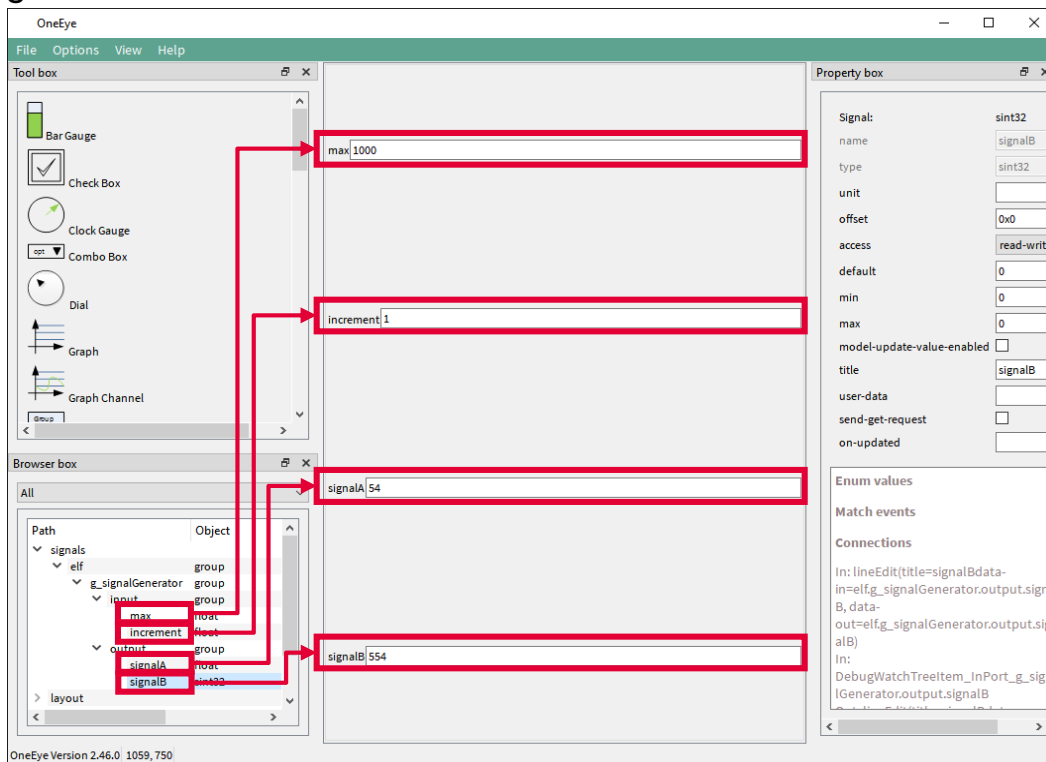
The created signals should appear in the browser box under the “signals.elf” group

# Implementation - OneEye

## Show and edit variables

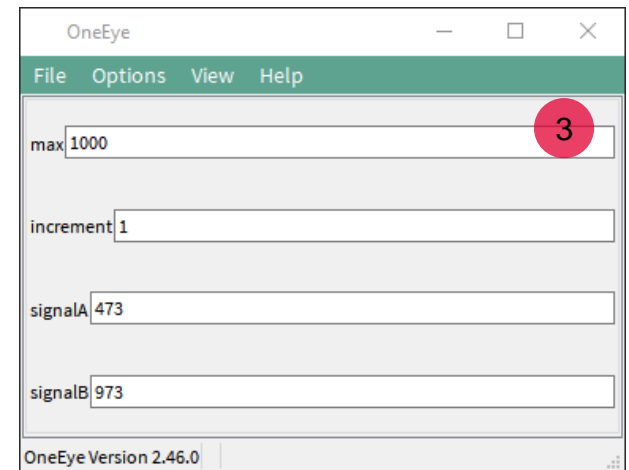
Drag and drop each of the variable corresponding signals (**max**, **increment**, **signalA**, **signalB**) from the browser box onto the layout to create default widget for them.

The values for **signalA** and **signalB** should be changing, if it is not the case check if the “**DAS connection status**” is checked in the debug box viewer. One can change the **max** and **increment** values to change the generator behaviour.



Save your configuration with CTRL+S.

Exit the edit mode with the OneEye menu “**Options -> Edit mode**” to only see the GUI **3**.



# References



- › AURIX™ Development Studio is available online:
- › <https://www.infineon.com/aurixdevelopmentstudio>
- › Use the „*Import...*“ function to get access to more code examples.



- › More code examples can be found on the GIT repository:
- › [https://github.com/Infineon/AURIX\\_code\\_examples](https://github.com/Infineon/AURIX_code_examples)



- › For additional trainings, visit our webpage:
- › <https://www.infineon.com/aurix-expert-training>



- › For questions and support, use the AURIX™ Forum:
- › <https://www.infineonforums.com/forums/13-Aurix-Forum>

## Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

**Edition 2022-03**

**Published by**

**Infineon Technologies AG  
81726 Munich, Germany**

**© 2021 Infineon Technologies AG.  
All Rights Reserved.**

**Do you have a question about this document?**

**Email: [erratum@infineon.com](mailto:erratum@infineon.com)**

**Document reference**

**OneEye\_DAS\_Quickstart\_1  
\_KIT\_TC334\_LK**

## IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”).

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer’s technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office ([www.infineon.com](http://www.infineon.com)).

## WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies’ products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.